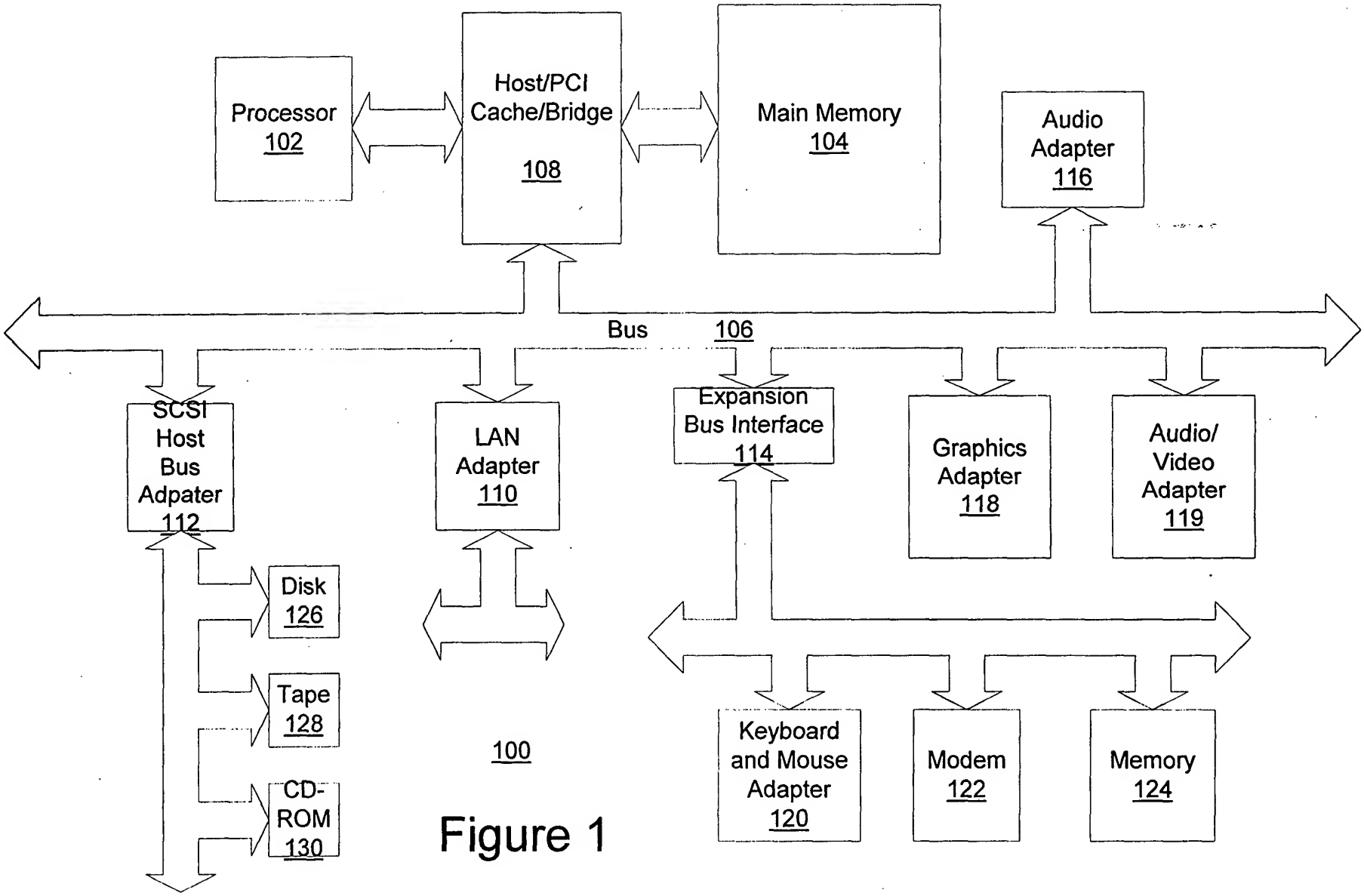

Processor
102
Host/PCI
Cache/Bridge
108
Main Memory
104
Audio
Adapter
116
Bus
106
SCSI
Host
Bus
Adpater
112
LAN
Adapter
110
Expansion
Bus Interface
114
Graphics
Adapter
118
Audio/
Video
Adapter
119
Disk
126
Tape
128
CD-
ROM
130
100
Keyboard
and Mouse
Adapter
120
Modem
122
Memory
124

Figure 1

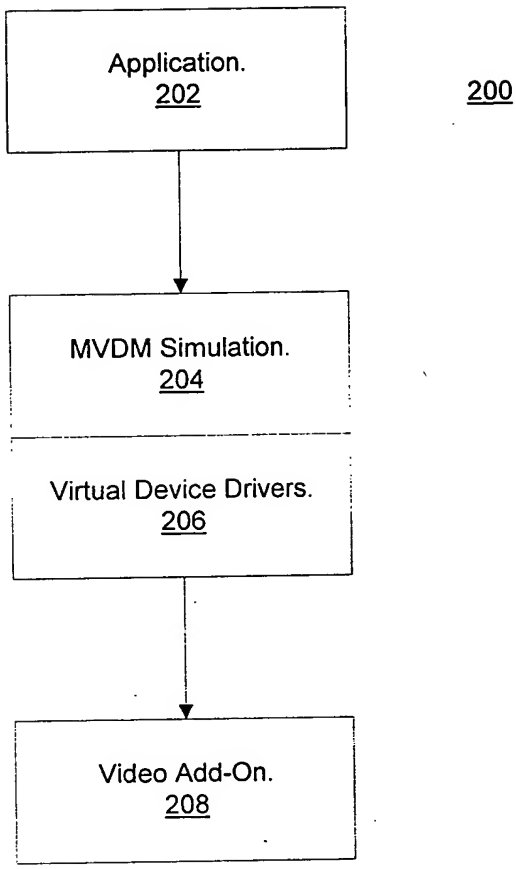


Figure 2

Figure 3

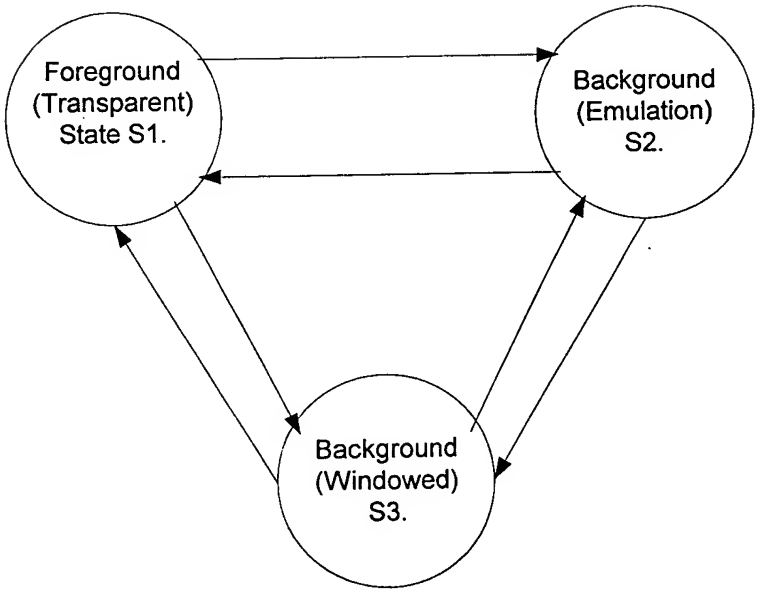
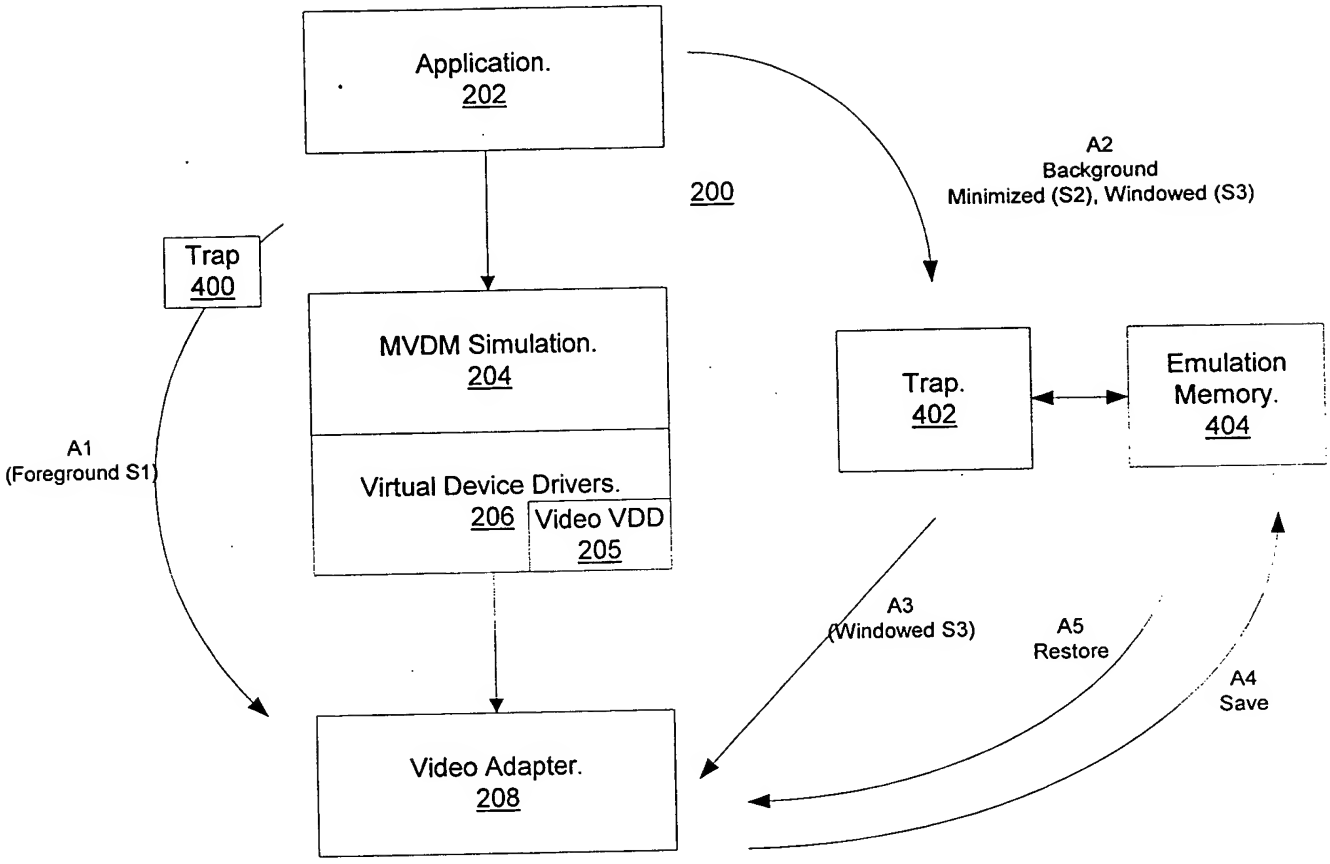


Figure 4


Application.
202
200
A2
Background
Minimized (S2), Windowed (S3)
Trap
400
MVDM Simulation.
204
Virtual Device Drivers.
206
Video VDD
205
Trap.
402
Emulation
Memory.
404
A1
(Foreground S1)
A3
(Windowed S3)
A5
Restore
A4
Save
Video Adapter.
208

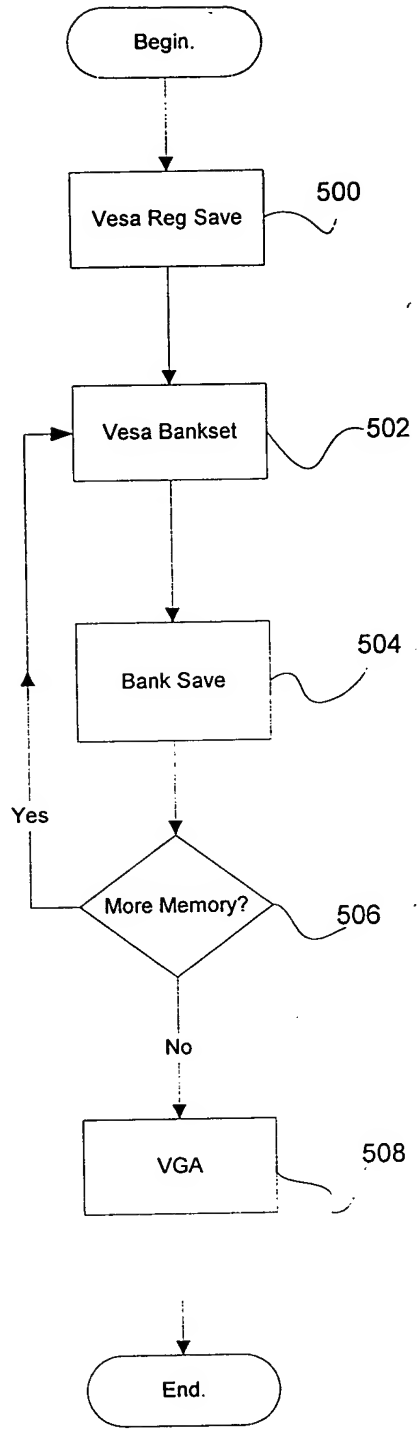
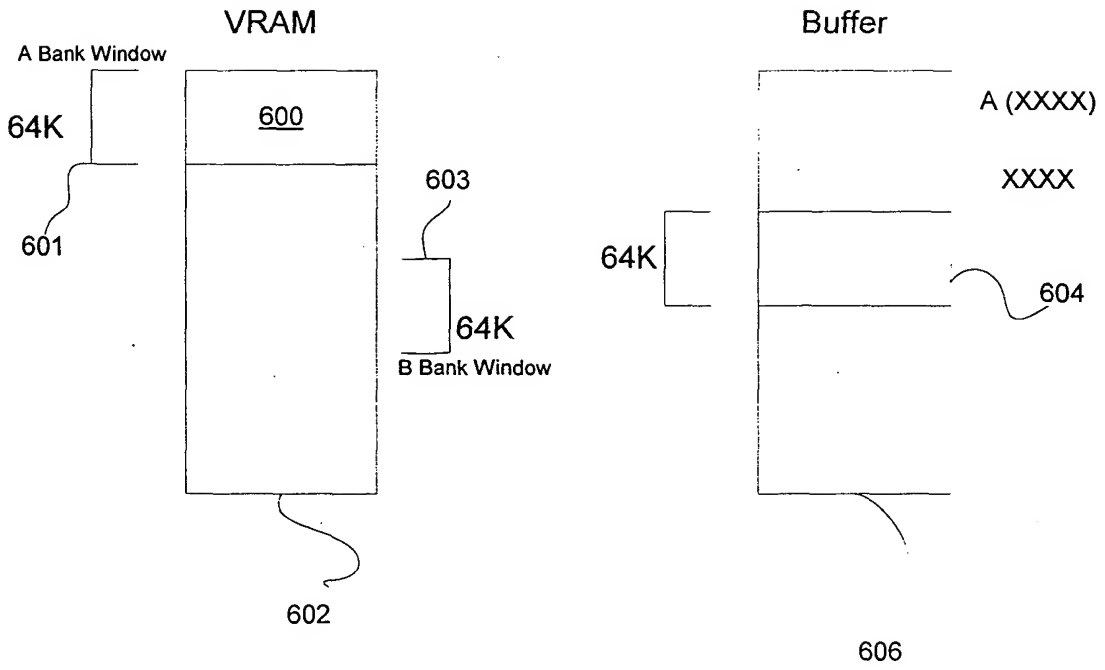

Begin.
Vesa Reg Save
500
Vesa Bankset
502
Bank Save
504
Yes
More Memory?
506
No
VGA
508
End.

Figure 5

Figure 6


VRAM
A Bank Window
64K
601
600
603
64K
B Bank Window
602
Buffer
A (XXXX)
XXXX
64K
604
606

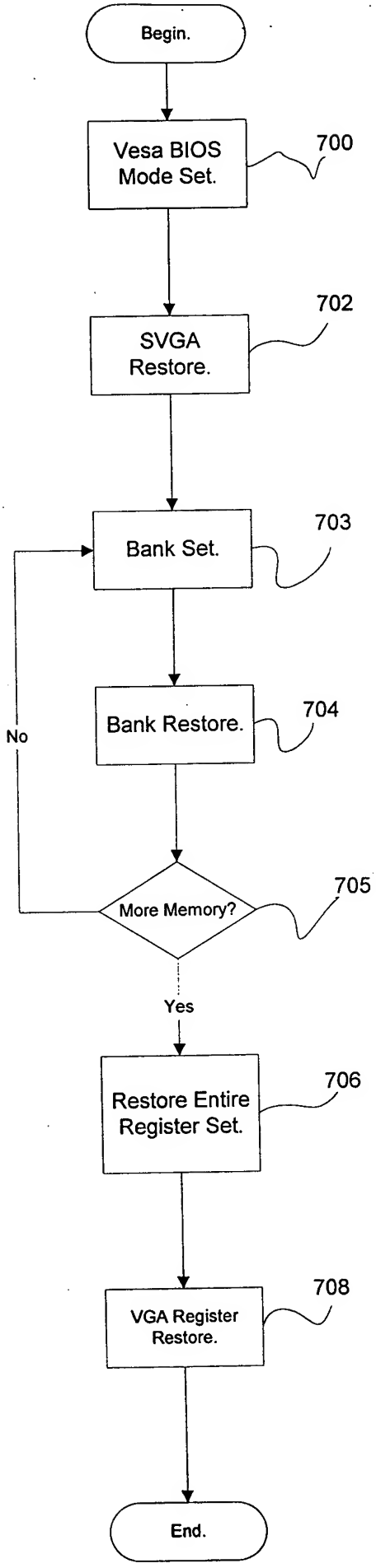

Begin.
Vesa BIOS Mode Set.
700
SVGA Restore.
702
Bank Set.
703
Bank Restore.
704
No
More Memory?
705
Yes
Restore Entire Register Set.
706
VGA Register Restore.
708
End.

Figure 7

```
/ ************************************************************************
 *
 * FUNCTION NAME = vvUserFgndS tMode
 *
 * DESCRIPTION
 *       Save client machine CPU r gister state
 *       Sav  video BIOS,data area
 *       Setup a VGA (or possibly VESA) BIOS call to set tne current
 *       client video mode in order to restore the VDM's state.

 ************************************************************************/

vvUserFgndSetMode()
{
  /* New art */
  Save client CPU register state
  Save video BIOS data area
  setup VGA (or possibly VESA) BIOS call to set the current
    client video mode
  return to
    vvUserFgndLogicalLineLength
}

/************************************************************************
 *
   FUNCTION NAME = vvUserFgndLogicalLineLength
 *
 * DESCRIPTION
 *       Setup a VESA BIOS call to set the logical scan line length
 *       Useful for VESA BIOS not implementing full register restore.
 *
 ************************************************************************/

vvUs rFgndLogicalLineLength()
{
  /* New art */
  inj ct vesa call to restore
  logical scan length start registers from saved area
  return to
      vvUserFgndDisplayStart,
}

/************************************************************************
 *
 * FUNCTION NAME = vvUserFgndDisplayStart
 *
 * DESCRIPTION
 *       Setup a VESA BIOS call to set the display start registers
 *       inserts the int 10 instruction, and arms a return to
 *       vvUserFgndBankCopy.
 *
 ************************************************************************/

vvUserFgndDisplayStart()
{
  /* New art */
  inject vesa call to restore display start registers from saved area
  return to
      vvUserFgndRegsS t,
}

/************************************ ****************** ***************
 *
```

800

802

804

Figure 8A

```
 * FUNCTION NAME = vv  rFgndRegsSet
 *
 * DESCRIPTION
 *      Setup a VESA BIOS call to restor  the clients adapter registers
 *
 ******* ************************************************************************/

vvUserFgndRegsSet()
{
  /* New art */
  inject vesa call to restore client adapter registers from saved area
  r turn to
      vvUserFgndBankSet1st,
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserFgndBankSet1st
 *
 * DESCRIPTION
 *      Setup a VESA BIOS call to set the VRAM bank number to 0;
 *
 *****************************************************************************/

vvUs rFgndBankSet1st()
{
  /* New art */
    if( Mode uses Linear Frame Buffer )
       transfer LINEAR buffer contents to VRAM from saved area
       inject vesa call to set A Bank to saved A bank
       return to
        vvUserFgndBankBSet,
    else
      pvd->VdmUser.lBankCopyNextBank = 0;
       inject vesa call to set A Bank to next bank # for restore
      return to
        vvUserFgndBankCopySetBBank,
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserFgndBankCopySetBBank
 *
 * DESCRIPTION
 *      Set the B Bank Window if it is needed for read/write operations.
 *      Most adapters only have an A Bank.
 *      A few have an A Bank for reading and a B Bank for writing,
 *      or vice versa.
 *
 *****************************************************************************/

vvUserFgndBankCopySetBBank()
{
  /* New art */
  inject vesa call to set B Bank to next bank # for restore
  return to
      vvUserFgndBankCopy,
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserFgndBankCopy
 *
 * DESCRIPTION
```

806

808

810

Figure 8B

```
 *      Transfers virtual memory to the VRAM bank,
 *      and then setup a VESA BIOS call to access    next A bank.
 *
 *      On the last pass, it does the transfer of virtual memory to the VRAM
 *      bank, and then setup a VESA BIOS call to set the bank
 *      number to the clients current A bank number.
 *
 ************************************************************************/

vvUserFgndBankCopy()
{
  /* Prior art */
  transfer one (current) bank of VRAM from saved area
  /* New art */
  increment bank number
  if( copy bank < total banks )
      inject vesa call to set A Bank to next bank # for restore
      return to
      vvUserFgndBankCopySetBBank,
   lse
      inject vesa call to set A Bank to client bank #
      return to
      vvUserFgndBankBSet,
}

/************************************************************************
 *
 * FUNCTION NAME = vvUserFgndBankBSet
 *
 * DESCRIPTION
 *      Setup a VESA BIOS call to set the B bank
 *      number to the clients current bank number.
 *      Most adapters only have an A Bank.
 *      A few have an A Bank for reading and a B Bank for writing,
 *      or vice versa.
 *      Useful for VESA BIOS not implementing full register restore.
 *
 ************************************************************************/

vvUserFgndBankBSet()
{
  inject vesa call to set B Bank to saved bank #
  return to
      vvUserFgndRegsSetAtEnd,
}

/************************************************************************
 *
 * FUNCTION NAME = vvUserFgndRegsSetAtEnd
 *
   DESCRIPTION
 *      Setup a VESA BIOS call to restore the client adapter
 *      register set to clean up the registers changed
 *      during the restoring the VRAM banks.
 *
 ************************************************************************/

vvUserFgndRegsSetAtEnd()
{
  inject vesa call to restore registers from saved state
  return to FgndFinish
}

/************************************************************************
```

812

814

816

Figure 8C

```
 *
 * FUNCTION NAME = vv. erFgndFinish
 *
 * DESCRIPTION
 *       Finish foreground switch in VDM's context.
 *       Restore the VGA register state directly.
 *       Useful for VESA BIOS not implementing full register restore.
 *       Restore client machine CPU register state sav d
 *       Restore video BIOS data area sav d
 *       Switch trapping behavior to transparent real hardware access.
 *
 ************************************************************************/

vvUserFgndFinish()
{
  /* Prior art */
  restore VGA register state
  /* New art */
  restore client machine CPU register state saved
  restore video BIOS data area saved
  /* Prior art */
  switch trapping behavior to transparent real hardware access
  thaw VDM when in unemulatable (SVGA) video mode.
}

/************************************************************************
 *
 * FUNCTION NAME = vvUserBgndSaveSizeQuery
 *
 * DESCRIPTION
 *       Save the VGA register state directly.
 *       Useful for VESA BIOS not implementing full register save.
 *       Save client machine CPU register state
 *       Save video BIOS data area
 *       Setup a VESA BIOS call to get the clients SVGA regs save area size.
 *
 ************************************************************************/

vvUserBgndSaveSizeQuery()
{
  /* New art */
  Save client machine CPU register state
  Save video BIOS data area
  inject VESA BIOS all to get client SVGA regs save area size
  return to
       vvUserBgndRegsGet,
}

/************************************************************************
 *
 * FUNCTION NAME = vvUserBgndRegsGet
 *
 * DESCRIPTION
 *       Checks the SVGA regs save area size returned.
 *       If the DOS allocated save area is large enough,
 *       then it issues the VESA BIOS call to save the SVGA registers.
 *       Setup a VESA BIOS call to save adapter register state.
 *
 ************************************************************************/

vvUserBgndRegsGet()
{
  /* New art */
  Setup a VESA BIOS call to save adapter register state.
```

818

904

902

Figure 9A

```
  return to
        vvUserBgndLo   alLineLength,
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserBgndLogicalLineLength
 *
 * DESCRIPTION
 *      Setup a VESA BIOS call to get the clients VRAM bank number.
 *
 *****************************************************************************/

vvUserBgndLogicalLineLength()
{
  /* New art */
  S tup a VESA BIOS call to get the clients VRAM bank number.
  return to
      vvUserBgndDisplayStart,
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserBgndDisplayStart
 *
 * DESCRIPTION
 *      Save returned logical line length values.
 *      Setup a VESA BIOS call to get the clients display start offset.
 *
 *****************************************************************************/

vvUserBgndDisplayStart()
{
  /* New art */
  Save returned logical line length values.
  Setup a VESA BIOS call to get the clients display start offset.
  return to
      vvUserBgndBankGet,
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserBgndBankGet
 *
 * DESCRIPTION
 *      Save returned display start values.
 *      Setup a VESA BIOS call to get the clients VRAM A bank number.
 *
 *****************************************************************************/

vvUserBgndBankGet()
{
  /* New art */
  Save returned display start values.
  S tup a VESA BIOS call to get the clients VRAM A bank number.
  return to
        vvUserBgndBankBGet,
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserBgndBankBGet
 *
 * DESCRIPTION
```

904

906

908

Figure 9B

```
 *       Save returned   bank number.
 *       Setup a VESA .  OS call to get the clients \  AM B bank number.
 *
 ********** ************************************************************/

vvUserBgndBankBGet()
{
  /* New art */
  Save return d A bank number.
  set current copy bank to -1
  Setup a VESA BIOS call to get the clients VRAM B bank number.
  return to
      vvUserBgndBankCopy
                                                          /*@V4.0JAN01*/
}

/*************************************************************************
 *
 * FUNCTION NAME = vvUserBgndBankCopy
 *
 * DESCRIPTION
 *       On the 1st pass,
 *         Save returned client B bank number.
 *         Setup a VESA BIOS call to set the VRAM bank number to 0.
 *
 *       On all middle passes,
 *         Tranfers the VRAM bank to virtual storage,
 *         Setup a VESA BIOS call to access the next VRAM bank.
 *
 *       On the last pass,
 *         Transfers the last VRAM bank to virtual storage,
 *         Setup a BIOS call to set VGA mode via vvUserBgndVGAModeSet
 *
 *************************************************************************/

vvUserBgndBankCopy()
{
  /* New art */
  if( copy bank < 0 )
    save returned client B bank number
  else
    /* Prior art */
    transfer one VRAM bank to saved area
  /* New art */
  if( mode uses Linear Frame Buffer )
    transfer whole linear buffer to save area
    return to
      vvUserBgndVGAModeSet
  else
    increment copy bank number
    if( copy bank number < total banks )
      setup a VESA call to set copy A bank number.
      return to
          vvUserBgndBankCopySetBBank,
    else
      call vvUserBgndVGAModeSet directly
}

/*************************************************************************
 *
 * FUNCTION NAME = vvUserBgndBankC pySetBBank
 *
 * DESCRIPTION
 *       Setup VESA BIOS call to set the copy B Bank Window,
 *       if it is needed for read/write operations.
```

910

912

914

Fig. 9C

```
 *
 ********************  ********************* *****  ************************/

vvUs rBgndBankCopySetBBank()
{
  /* New art */
  Setup VESA BIOS call to set the copy B Bank Window,
  return to
    vvUserBgndBankCopy
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserBgndVGAModeSet
 *
 * DESCRIPTION
 *      Setup a VGA BIOS call to set a VGA standard video mode (mode 12).
 *      This allows next operating system component manipulating the
 *      video hardware to assume the SVGA is a simple/standard VGA.
 *
 *****************************************************************************/

vvUserBgndVGAModeSet()
{
  /* New art */
  setup a VGA BIOS call to set a VGA standard video mode.
  return to vvUserBgndFinish
                                                          /*@V4.0JAN01*/
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvUserBgndFinish
 *
 * DESCRIPTION
 *      Finish background switch in VDM's context
 *      Freeze VDM when in unemulatable (SVGA) video mode.
 *      Leave emulateable (VGA) video mode unfrozen.
 *
 *****************************************************************************/

vvUserBgndFinish()
{
  /* New art */
  restore client CPU register state
  /* Prior art */
  switch trapping behavior to emulation of hardware access
  freeze VDM when in unemulatable (SVGA) video mode.
}
```

916

918

920

Figure 9D

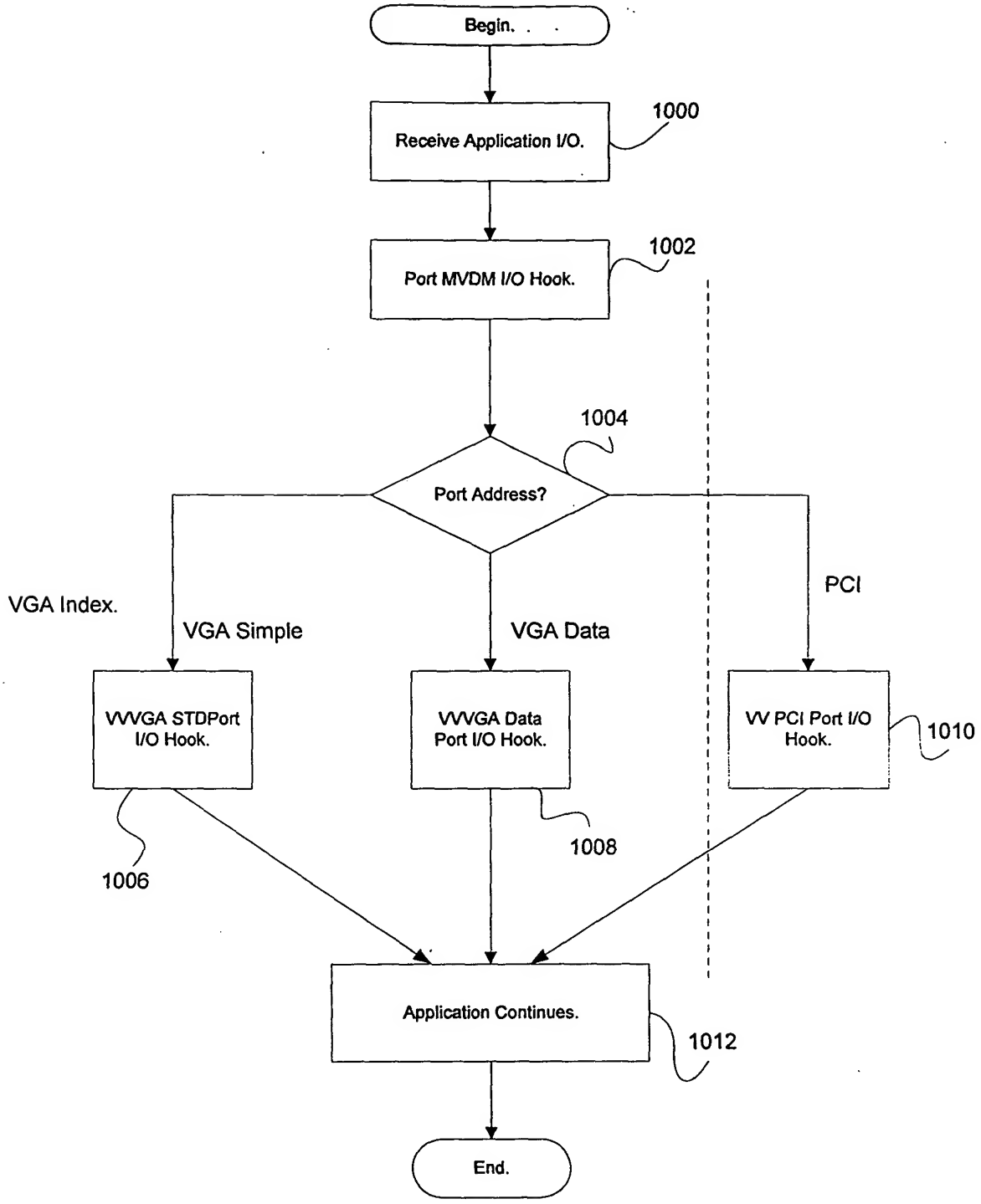

Begin.
Receive Application I/O.
1000
Port MVDM I/O Hook.
1002
Port Address?
1004
VGA Index.
VGA Simple
VGA Data
PCI
VVVGA STDPort I/O Hook.
VVVGA Data Port I/O Hook.
VV PCI Port I/O Hook.
1010
1006
1008
Application Continues.
1012
End.

Figure 10

```
/******* ************  ** ************************  ************************
 *
 * FUNCTION NAME = vvInit
 *
 * DESCRIPTION
 *      Initialization for virtual video driv r
 *      called by mvdm at start of each VDM
 *      Most VESA BIOSes now provide PCI BIOS information too
 *
 ****************************************************************************/

vvInit()
{                                                                    1100
  /* Prior art: */
  register standard VGA I/O port address handlers with mvdm.
  /* New art: */
  make PCI BIOS call to get list of PCI BIOS I/O port addresses.
  for each PCI BIOS I/O port address
    register PCI BIOS I/O port address handler with mvdm.
}

;/***************************************************************************
;*
;* FUNCTION NAME = mvdmIOHook
;*
;* DESCRIPTION
;*      All client I/O instructions generate a hardware trap which comes here
;*      Handlers are generally all registered at the start of the VDM.
;*      Video port hooking is enabled in the background,
;*      and disabled in the foreground.
;*      Non-video hardware follows other algorithms based on the
;*      device driver requirements and sophistication.
;*
;***************************************************************************/

mvdmPortIOHook()
{                                                                    1102
  /* All prior art */
  if( registered handler for I/O port address
      && hooking enabled for I/O port address )
     call registered handler for I/O port address
  else
     do I/O directly.
}

/***************************************************************************
 *
 * FUNCTION NAME = vvVGAStandardPortIOHook
 *
 * DESCRIPTION
 *      Typical registered hook handler for VGA Standard I/O port address
 *      May be more complicated if I/O port not connected to a simple register
 *      Such as pair of I/O ports for an index and data register array
 *      Each I/O port address may have its own unique and differently
 *      coded handler to handle unusually behaving ports.
 *
 ****************************************************************************/

vvVGAStandardPortIOHook()
{
  /* All prior art */                                                1104
  if( input )
    return ( emulation state variable value for I/O port address )
```

Fig. 11 A

```
    /* This go s into  he client CPU r gister set *
   Is  /* output */
    Save output from client CPU regist r set
    into  mulation state variable for I/O port address
    /* Will be used later to r store adapter contents */
    Adjust any other emulation state variables required by changes to this port
}

/************************************************************************ **********
 *
 * FUNCTION NAME = vvVGADataPortIOHook
 *
 * DESCRIPTION
 *      Typical registered hook handler for VGA Data I/O port address
 *      as a part of index and data port handler pair.
 *      Index port handler is usually a vvVGAStandardPortIOHook.
 *
 ************************************************************************/

vvVGADataPortIOHook()
{
  /* All prior art */
  if( input )
    return ( emulation state variable [index port state variable]
             value for I/O port address )
    /* This goes into the client CPU register set */
  else /* output */
    Save output from client CPU register set
    into emulation state variable [index port state variable]
      for I/O port address
    /* Will be used later to restore adapter contents */
    Adjust any other emulation state variables required by changes to this port
}

/************************************************************************
 *
 * FUNCTION NAME = vvPCIPortIOHook
 *
 * DESCRIPTION
 *      Registered by the virtual video device driver for a list
 *      of port addresses provided by the PCI BIOS.
 *      ONLY registered hook handler type for PCI BIOS I/O port address.
 *      This represents a simple best guess to how a typical port works.
 *      But it often does not absolutely correct emulation.
 *      However it almost always suffices for emulating VGA modes.
 *      This is NOT true of SVGA modes,
 *      and this is why we freeze when in VESA modes in the background
 *      so that the video adapter is not incorrectly emulated.
 *      Emulation state variables used here
 *      will NOT be used later to restore adapter contents,
 *      because we do not know how port really works!
 *      Instead we rely on the VESA BIOS calls to restore important registers.
 *
 ************************************************************************/

vvPCIPortIOHook()
{
  /* New art */
  if( input )
    return ( emulation state variable value for I/O port address )
    /* This g es into the client CPU register set */
  else /* output */
    save output from client CPU register set
    into emulation state variable for I/O port address
```

1104

1106

1108

Fig 11B

```
;/*******************  *************  ************  ************************
;*
;* FUNCTION NAME = vvInt10PreHook
;*
;* DESCRIPTION
;*       Quick Return if not Mode Set,
;*       else tranfer contr l to
;*
;**** ***************************************************************/

vvInt10PreHook()
{
  if( AH( pcrf ) == 0x00)    /* VGA Mode Set */
  {
    /* Prior art */
    save client registers as last setmode registers
    vvInt10Chain
  }
  else if( AX( pcrf ) == 0x4F02 )  /* VESA Mode Set */
  {
    /* Prior art */
    save client registers as last setmode registers
    /* From here begins new art: */
    save VESA setmode number
    push client registers
    inject VESA call to get VESA BIOS SVGA INFO.
    return to vvInt10VesaVbeInfoReturn
  }
  else
    /* Prior art */
    vvInt10Chain
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvInt10VesaVbeInfoReturn
 *
 * DESCRIPTION
 *       Sets up for a VESA Mode query.
 *
 *****************************************************************************/

vvInt10VesaVbeInfoReturn()
{
  save VESA BIOS SVGA INFO including total VRAM size.
  inject VESA BIOS call to get MODE INFO for new mode.
  return to
       vvInt10VesaModeInfoReturn
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvInt10VesaModeInfoReturn
 *
 * DESCRIPTION
 *       Gets the VESA mode information from the Mode information block and
 *       copies it to the VDM's VESA mode information structure, and then
 *       sets up to do the actual VESA BIOS mode set to the VESA mode.
 *
 *****************************************************************************/

VOID HOOKENTRY vvInt10VesaModeInfoReturn()
  PSZ pcrf;
```

1200

1202

1204

Figure 12A

```
{
  pop client register
  save current mode inro as old mode info
  save VESA BIOS MODE INFO as current mode info
    ( includes mode dimension info )
  inject original setmode call to original VESA BIOS INT 10 handler
  return to
    vvInt10V saEndReturn
}

/*************************************************** ***************************
 *
 * FUNCTION NAME = vvInt10VesaEndReturn
 *
 * DESCRIPTION
 *      Does the post cleanup after the VESA BIOS mode set.
 *
 ******************************************************************************/

vvInt10VesaEndReturn()
{
  if( AX( pcrf ) != VESA_FUNCTION_SUCCESS )
    restore current mode info from old mode info
  else if( background )
    freeze VDM
  vvInt10Continue
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvInt10Chain
 *
 * DESCRIPTION
 *      Continue with client INT 10
 *
 *****************************************************************************/

vvInt10Chain()
{
  call original (VGA/VESA) BIOS INT 10 handler
  return to vvInt10Continue
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvInt10Continue
 *
 * DESCRIPTION
 *      return to client program
 *
 *****************************************************************************/

vvInt10Continue()
{
  return to client program
}

/*****************************************************************************
 *
 * FUNCTION NAME = vvDsvModeUpdate
 *
 * DESCRIPTION
 *      Determine current mode dimensions
 *      Thes  dimensions ar  us d to determine:
```

Figure 12B

```
 *       A) How much V  AM to sav  and restore for e  lation switching
 *       B) How to dr   current VRAM contents as a   cture in a desktop window
 *
 ************************************************************************/

vvDsvModeUpdate()
{
  if ( VESA MODE )
    /* New art: */
    calculate mode dimensions from info returned by previous VESA calls
      (current mode info)
  else
    /* Prior art */
    calculate mode dimensions from standard VGA registers
}
```

12/14

Figure 12C